DUKASCOPY
Swiss Forex Bank & Marketplace

ChatBot API: An introduction	
Subject	**Dukascopy Connect 911 ChatBot API**
Last update	30 August 2018

DUKASCOPY
Swiss Forex Bank & Marketplace

Contents

Revision control section

Date	Version	Summary of changes made
18.04.2018	V0.1	Initial Release
12.09.2018	V1.0	Public Release

Abbreviations and definitions

Dukascopy Connect 911	Dukascopy Connect 911 is a complex cross platform client-server messenger application that focuses on ensuring swift and secure message delivery between employees and clients of Dukascopy Bank.
WebSocket	Advanced technology that makes it possible to open an interactive communication session between the user's browser and a server. With websockets, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.
Chat Bot	A computer program that simulates human conversation, or chat, through artificial intelligence. Typically, a chat bot will communicate with a real person.
Invoice	Payment request from a user to user. Payment currently is possible from only registered users in Dukascopy Bank.

1. Introduction to bots

Bots are software extensions that use Dukascopy Connect 911 as a method of communication between the user and 3rd party software. Dukascopy Connect users will be able to interact with your Chatbot by sending those messages, commands and bot specific requests. All bots utilize bot API to connect and work with Dukascopy Connect 911 Messaging platform.

2. What can a bot do?

Few examples of what a bot can do but are not limited:

- Provide Mobile Banking Services – Dukascopy provides an example of Bank bot to allow you to have full control over your multi-currency account through a Chatbot interface.
- Integrate with other services – Bot can improve chats with content from external services (Translate Bot)
- Accept payments from Dukascopy Connect 911 users – (Pizza bot)

3. How do bots operate?

Bots in Dukascopy connect are special accounts that do not require a special phone number to operate as they are linked to their respective master account.

The bots can only be registered if the person applying for the bot has an active account with Dukascopy Bank.

You can find bots that are operational in Dukascopy in special Chatbot section. Chatbot's cannot start chats on their own, but they can be added to private chats or to group chats and public channels.

Bots in Dukascopy Connect 911 operate through a special ChatBot API that allows easy, secure and transparent way on how to connect to the Dukascopy Connect 911 messenger platform.

4. How to create a new bot

To create a bot first of all you need to obtain an authorization token. To obtain authorization token you must have an account in Dukascopy Bank SA. After you have successfully opened an account in Dukascopy Bank SA you need to contact special Bot named BotFactory that will ask you questions about your new bot. BotFactory will issue a new authorization token. Each Dukascopy account owner can create 3 bots, if you need more please contact us directly via email 911@dukascopy.com

After you answer all of the questions posed by the BotFactory you will receive authorization key that looks like "c4a213e4dwqcc1b229e8497f413ds3319f923".

After receiving an authorization token you can proceed with bot usage as per Bot API.

5. Bot API

Dukascopy Connect 911 BOT API operates through a WebSocket connection. Both the commands and responses from BOT API are sent in JSON format.

a. Connecting

To connect to Chatbot API open WebSocket connection to ws://bot-api.telefision.com port 80 or you can use a secure connection and connect to wss://bot-api.telefision.com port 443.

b. Authorisation

After you successfully open connection within 5 seconds you need to authorize in the BOT API by sending data with the following payload:

```
{"method": "auth",
"authKey":"a7a263e2fwqcc1b229e8497f413ds3319f925"}
```

If you have successfully authorized in the ChatBot API you will receive the answer from the server:

```
{"method":"init","userUID":"kcxsS2"}
```

Where in data you will see the unique user identification of your chat bot, you should use this unique identification to filter messages from the Chatbot API.

c. Ping-Pong routine.

The Websocket server will send ping command every 10 seconds. The client should respond with pong command within 15 seconds from the last ping otherwise the client will be disconnected.

```
{method:"ping"}
```

Therefore as soon as your see ping method in your socket connection your Chatbot should respond with pong method.

```
{method:"pong"}
```

d. Chat Started

As soon as the chat is started with your bot. You will receive a start command which will additionally contain information about the user device language and username. This will allow to respond in a language which user is best to understand.

The data in Websocket frame will look as following:

```
{method: "botCommand", data: { action:"start"}, actionData:
{lang:"en", userName:"ChatBot user"}}
```

The language will be sent as in ISO 639-1 standard and Dukascopy Connect at the current moment supports following interface languages.

Language	ISO-639-1 Code
English	en
Russian	ru

French	fr
Italian	it
Japanese	jp
Polish	pl
Slovak	sk
Hungarian	hu
Chinese	zh

e. Listen for messages

If the bot is added to chat all events happening in the chats will be relayed through the Chatbot API.

A message that has been sent to the chat will be displayed in the following format:

- Method – addMsg
- chatUID - Chat to which the menu needs to be sent.
- userUID (optional) - user to which the menu needs to be sent, in private chats this parameter can be ignored. In group chats this parameter allows menu to be sent in private.
- msgId - message identifier
- msgNum - message number in the chat
- created - timestamp of the created message

- text - text of the message (limited to 2000 characters)

```
{method:"addMsg", chatUID:"WS123", "userUID":"user123",
msgId:23665, msgNum:3, created:1536077382, text:"Message from
User"}
```

Where chatUID will be Unique Chat ID, userUID will be unique ID of the user that has sent the message and the text field will represent the text message. Each time a bot is added to a new chat bot will start receiving messages from them.

f. Sending a message

To send a message to the chat of your choice you must use addMsg method. Please note that your bot must be a member of the chat and for channels should have sufficient permissions to post.

```
{ method: "addMsg", chatUID: "123", text: "Sample text here" }
```

Where chatUID will be the chat where to send the message and and text will be will represent the text of the message.
After you send a message it will be relayed back to you, therefore it is important that you ignore the messages sent with userUID that is equal to the UID of your Chabot. The addMsg method text field is limited to 2000 symbols.

g. Sending and accepting payments.

It is possible to request funds from your chat bot users for services. In order to send an invoice the request you should use addInvoice method and you will receive any updates for your

invoice through updateInvoice method. the owner of the Chatbot will be

To send an invoice your request should contain the following data

- Method – addInvoice
- chatUID - Chat to which the menu needs to be sent.
- invoice - Object containing below elements
    - currency - Currency in which invoice is issued. Can be in one of the supported currencies. EUR, CHF, USD, GBP, AUD, CAD, PLN, RUB, CNH, DKK, SEK, NOK, SGD, HKD, MXN, NZD, TRY, ZAR, CZK, HUF, ILS, RON
    - amount - Invoiced amount.
    - toUserUID - Unique user identifier to which the invoice is sent.
    - purposeMsg (optional) - Optional message that will be displayed for the end user

```
{
method:"addInvoice",
chatUID:"WMWeDMWOWxW2WB",
invoice:{
     currency:"EUR",
     amount:0.5,
     toUserUID:"WGWKDaWzWXZwIi",
     purposeMsg:"invoice details, any text up to 200 symbols"
(optional)
}
}
```

As soon as there is any user interaction with the sent invoice you will receive an update through Websocket. the update will be issued through the updateInvoice method:

```
{
```

```
method:"updateInvoice",
chatUID:"chat uid",
invoice: {
     currency:"EUR",
     amount:0.5,
     toUserUID:"user uid",
     purposeMsg:"details if were sent"
     status:"payed" or "cancel" or "rejected"

}}
```

h. Sending a custom menu

As an alternative to sending plain text messages it is possible to elaborate user interfaces in the chatbot. In order to send a menu you need to submit a request with the following structure.

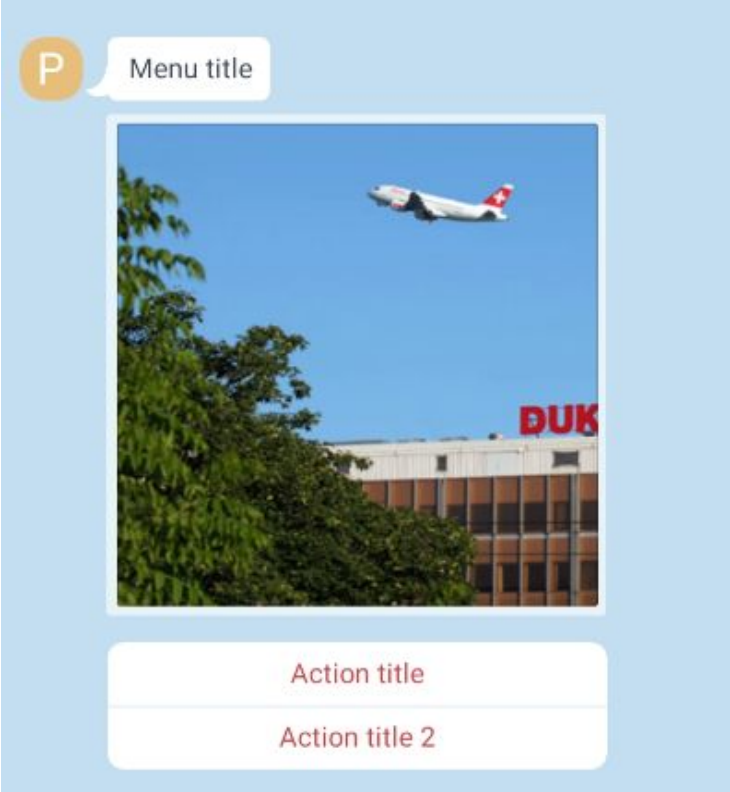
- Method – addMenu
- chatUID - Chat to which the menu needs to be sent.
- userUID (optional) - user to which the menu needs to be sent, in private chats this parameter can be ignored. In group chats this parameter allows menu to be sent in private.
- Menu:
    - Title – (String) Text that will be shown before the menu (optional)
    - Image (URL) – Link for the image that will be showed before the menu (optional)
    - Items (Array) – Array of menu elements
        - Text (String) - Text label that will represent a menu element.

- Action (String) - Command that will be sent to the bot.
- actionData (String) (optional) -
- displayText (String) (optional) - Display text will be displayed to the end user.

Example of the menu is as following:

```
{
	"method": "addMenu",
	"chatUID": "WMWeDMWOWxW2WB",
	"userUID": "WGWKDaWzWXZwIi",
	"menu": {
		"title": "Menu title",
		"image":
"https://www.dukascopy.com/media/Image/offices/banners_jet_sharp.jpg"
,
		"items": [{
			"text": "Action title",
			"action": "bot:bot action1"
		}, {
			"text": "Action title 2",
			"action": " bot:bot action2"
		}]
	}
}
```

The example menu will result in displayed menu which will have title "Menu title" one picture, and two possible actions.



When the user selects "Action title" element the following message will be sent to the chat bot:funbot action1. And in the websocket connection the message will look as following.

```
{ method: "addMsg", chatUID: "123", userUID: "234", text: "bot: bot action1" }
```

This should make it possible to create complex Bot to User interactions.

6. Bot examples

Please feel free to look over our bot examples to get more

detailed insights on how bots can be used to improve your everyday life.

You can see the following Please see examples of created bots to learn more about their functionality:

@PizzaBot – a demonstration bot that features a shopping cart experience functionality. You can add/remove goods from and to your shopping basket, ask for invoice, pay the invoice.

@TranslateBot – a demonstration of 3rd party service integration. It is possible to use translate bot to

@WeatherBot – a demonstration of 3rd party service integration. It is possible to use weather bot to get the latest weather in all cities where branches of Dukascopy Bank are available.